LAB 08

STACK, IT'S OPERATION AND NESTED PROCEDURES

Syed Muhammad Faheem
STUDENT NAME

20K-1054
ROLL NO

3E
SEC

SIGNATURE & DATE

MARKS AWARDED: ______________

EL 2003 COMP ORG & ASSEMBLY LANGUAGE LAB

LABORATORY MANUAL •Fall 2021 •Instructor: Aamir Ali , Aashir Mehboob, Amin Sadiq, Qurat-ul-Ain, M Kariz, Rabia

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), KARACHI

Prepared by: Aashir Mahboob

Version: 1.0

Date: 27th Oct 2021

Lab Session 08: STACK,IT'S OPERATION & NESTED PROCEDURES

Objectives :

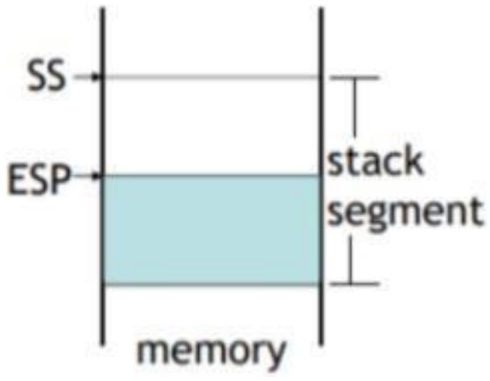
- To learn about Runtime Stack and how to implement using PUSH and POP instructions
- To learn about user defined procedures and to use related Instructions
- Undersatnding the Nested Procedures and the way those are implemented in assembly

Stack:

- LIFO (Last-In, First-Out) data structure.
- push/ pop operations
- You probably have had experiences on implementing it in high-level languages.
- Here, we concentrate on runtime stack, directly supported by hardware in the CPU. It is essential for calling and returning from procedures.

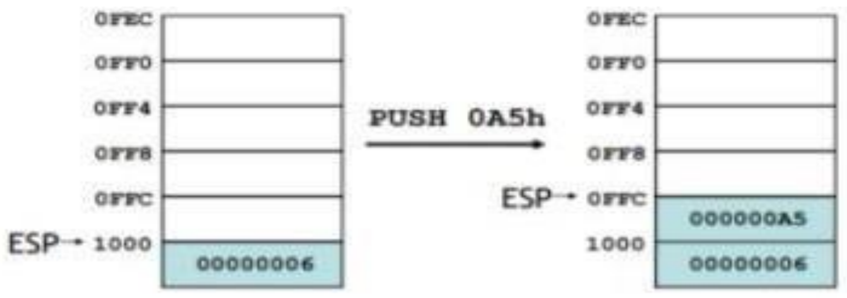
Runtime Stack:

- Managed by the CPU, using two registers
- SS (stack segment)
- ESP (stack pointer): point the last value to be added to, or pushed on, the top of stack usually modified by instructions: ***CALL, RET, PUSH and POP***

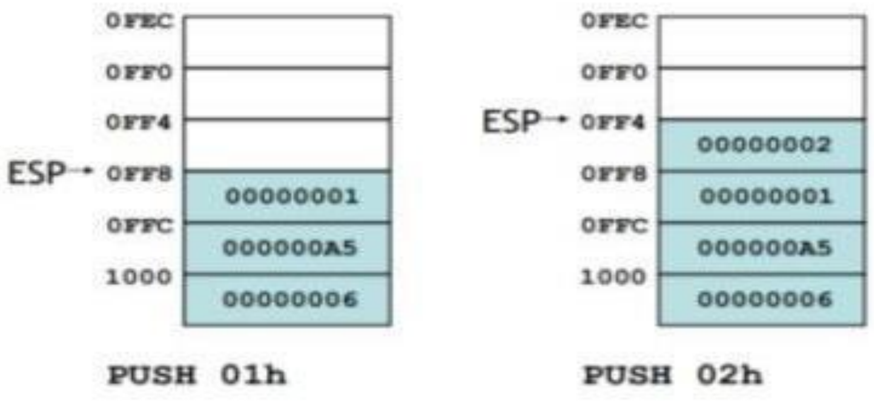


Push Operation

A 32-bit push operation decrements the stack pointer by 4 and copies a value into the location in the stack pointed to by the stack pointer.

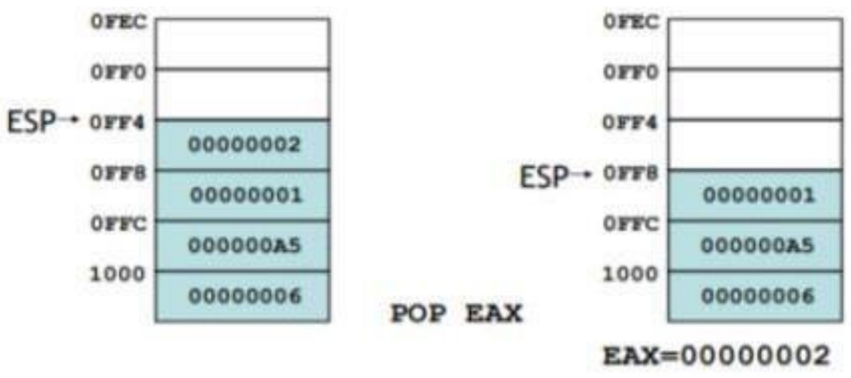


- The same stack after pushing two more integers:



Pop Operation

A pop operation removes a value from the stack. After the value is popped from the stack, the stack pointer is incremented (by the stack element size) to point to the next- highest location in the stack. It copies value at stack [ESP] into a register or variable.



PUSH and POP instructions:

PUSH syntax:

- PUSH r/m16
- PUSH r/m32
- PUSH imm32

POP syntax:

- POP r/m16
- POP r/m32

PUSHFD and POPFD Instructions

The MOV instruction cannot be used to copy the flags to a variable.

The **PUSHFD** instruction pushes the 32-bit EFLAGS register on the stack, and **POPFD** pops the stack into EFLAGS: PUSHFD
 POPFD

Example 01: (Stack and nested loops.)

```
        Include Irvine32.inc
     .code   main
proc
mov ecx,5
L1: push ecx
            mov ecx, 10
            L2:
                    inc ebx
             loop L2
            pop ecx
```

```
loop L1

call      DumpRegs exit
main ENDP
END main
```

<u>Example 02:(displays the Addition of three integers through a stack)</u>

```
Include Irvine32.inc
 .data
       VAR1 DWORD 2
.code main
proc
                mov eax, 0
       mov ecx, 3
                L1:
                          PUSH VAR1
  ADD VAR1, 2   LOOP L1
                mov ecx, 3
                L2:
                          POP ebx
                          ADD eax, ebx          ;eax value added
                LOOP L2

call DumpRegs exit
main ENDP
END main
```

<u>Example 03:(To find the largest number through a stack)</u>

```
Include Irvine32.inc
.code
main proc
PUSH 5
PUSH 7
PUSH 3
PUSH 2
MOV eax, 0                                        ;eax is the largest
MOV ecx, 4
L1:
                POP edx
                CMP edx, eax
```

```
                    JL SET
                    MOV eax, edx SET:
        LOOP L1
call    DumpRegs
        exit
        main ENDP
      END main
```

Procedures

- Procedures or subroutines are very important in assembly language, as the assembly language programs tend to be large in size.
- Procedures are identified by a name. Following this name, the body of the procedure is described which performs a well-defined job.
- End of the procedure is indicated by a return statement.

Example 04:

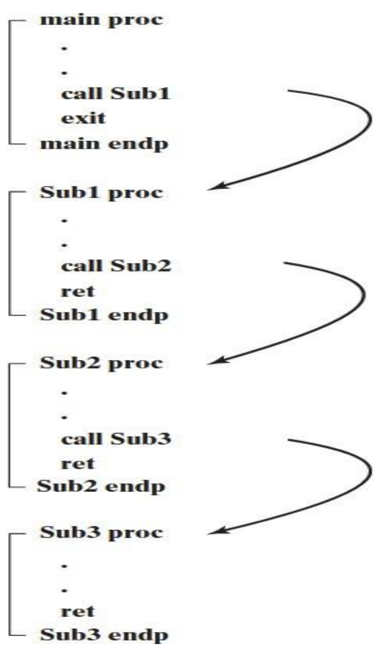
```
INCLUDE Irvine32.inc
INTEGER_COUNT = 3
.data
  str1 BYTE "Enter a signed integer: ",0
  str2 BYTE "The sum of the integers is: ",0
  array DWORD INTEGER_COUNT DUP(?)

.code
main PROC
call Clrscr
mov esi, OFFSET array   mov
ecx, INTEGER_COUNT   call
PromptForIntegers
call ArraySum
call DisplaySum


exit
main ENDP

   ;----------- PromptForIntegers -------------
PromptForIntegers PROC USES ecx edx esi mov edx,
OFFSET str1                    ; "Enter a signed integer"
```

```
L1:
        WriteString                 ; display string
        call ReadInt                ; read integer into EAX
        call Crlf                   ; go to next output line
        mov [esi], eax              ; store in array
        add esi, TYPE DWORD         ; next integer
loop L1
ret
PromptForIntegers ENDP

;----------- ArraySum -------------
ArraySum PROC USES esi ecx mov eax,0                                  ;
initialize the value of sum to ZERO
 L1:
        add eax, [esi]              ; add each integer to sum
        add esi, TYPE DWORD         ; point to next integer
loop L1                             ; repeat for array size ret
                                ; sum is in EAX
ArraySum ENDP

;----------- DisplaySum -------------
DisplaySum PROC USES edx  mov
edx, OFFSET str2
call WriteString
call WriteInt                       ; display EAX
call Crlf
ret
DisplaySum ENDP
 END main
```

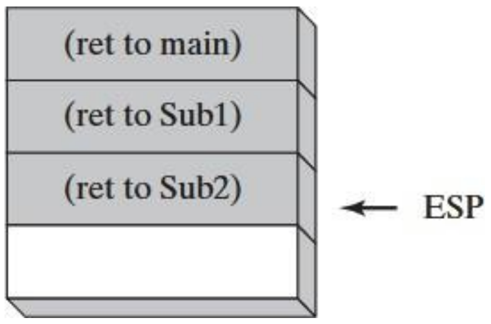
Nested Procedure Calls

A nested procedure call occurs when a called procedure calls another procedure before the first procedure returns.

.



```
main proc
  .
  .
  call Sub1
  exit
main endp

Sub1 proc
  .
  .
  call Sub2
  ret
Sub1 endp

Sub2 proc
  .
  .
  call Sub3
  ret
Sub2 endp

Sub3 proc
  .
  .
  ret
Sub3 endp
```



Example 05:

```
Include Irvine32.inc
 .data
       var1     DWORD 5
       var2     DWORD 6

.code      main
proc           call
AddTwo   call
dumpregs call
writeint     call
crlf
exit
main ENDP

AddTwo PROC
Mov eax,var1 Mov
ebx,var2
Add eax,var2

Call AddTwo1
Ret
Addtwo ENDP

AddTwo1 PROC
Mov ecx,var1
Mov edx,var2 Add
ecx, var2
Call writeint
Ret
AddTwo1 ENDP
```

Lab Task(s):

Task#1:

Take an array atleast of 10 numbers, move word-type of data in reverse order into

another empty array using stack push and pop technique.

Task#2

Write a program having nested procedures used to calculate the total sum of 2 arrays (each array having atleast 5-elements). The sum of 1-array in 1st procedure and in 2nd procedure have sum of 2-array. And the 3rd procedure adds the results of both.

Task#3

Print the following pattern using a function call in which number of columns is passed through a variable.

```
    *
   **
  ***
 ****
*****
```

Task#4

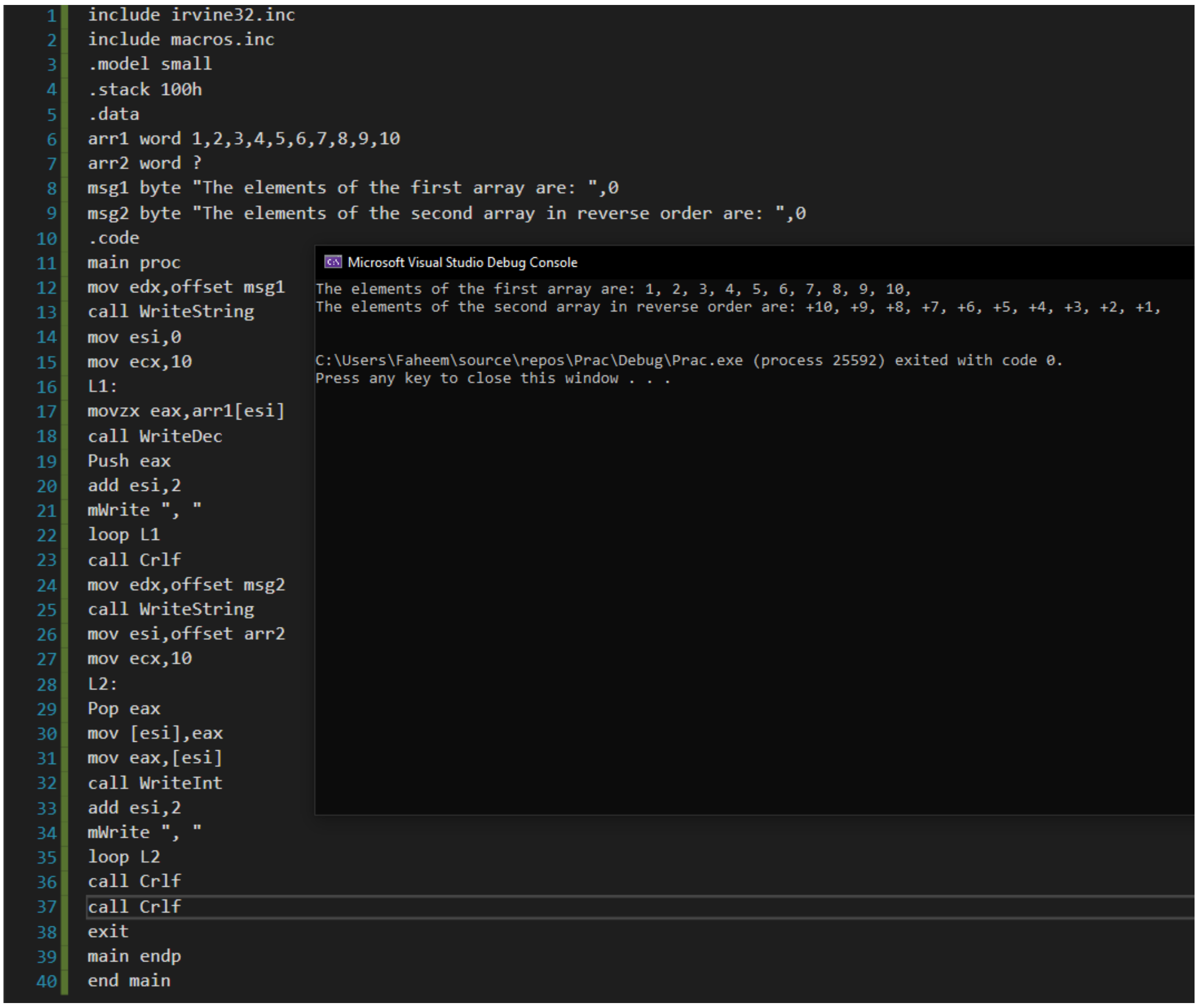
Print the following pattern using a function call in which number of columns is passed through a variable.

```
    A
   BC
  DEF
 GHIJ
KLMN
```

Task#5

Write a function that asks the user for a number n and prints the sum of the numbers 1 to n.

Task 1:

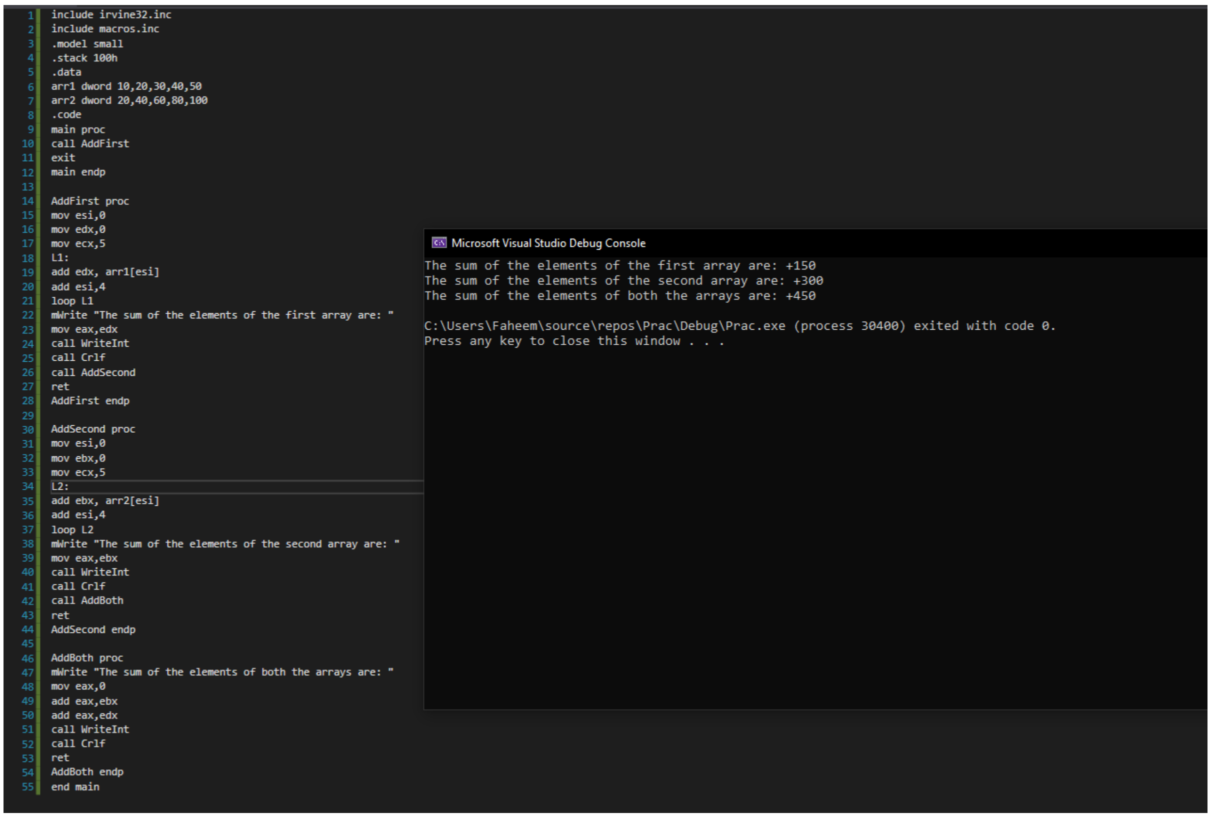


```
include irvine32.inc
include macros.inc
.model small
.stack 100h
.data
arr1 word 1,2,3,4,5,6,7,8,9,10
arr2 word ?
msg1 byte "The elements of the first array are: ",0
msg2 byte "The elements of the second array in reverse order are: ",0
.code
main proc
mov edx,offset msg1
call WriteString
mov esi,0
mov ecx,10
L1:
movzx eax,arr1[esi]
call WriteDec
Push eax
add esi,2
mWrite ", "
loop L1
call Crlf
mov edx,offset msg2
call WriteString
mov esi,offset arr2
mov ecx,10
L2:
Pop eax
mov [esi],eax
mov eax,[esi]
call WriteInt
add esi,2
mWrite ", "
loop L2
call Crlf
call Crlf
exit
main endp
end main
```

Microsoft Visual Studio Debug Console

```
The elements of the first array are: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
The elements of the second array in reverse order are: +10, +9, +8, +7, +6, +5, +4, +3, +2, +1,


C:\Users\Faheem\source\repos\Prac\Debug\Prac.exe (process 25592) exited with code 0.
Press any key to close this window . . .
```

Task 2:

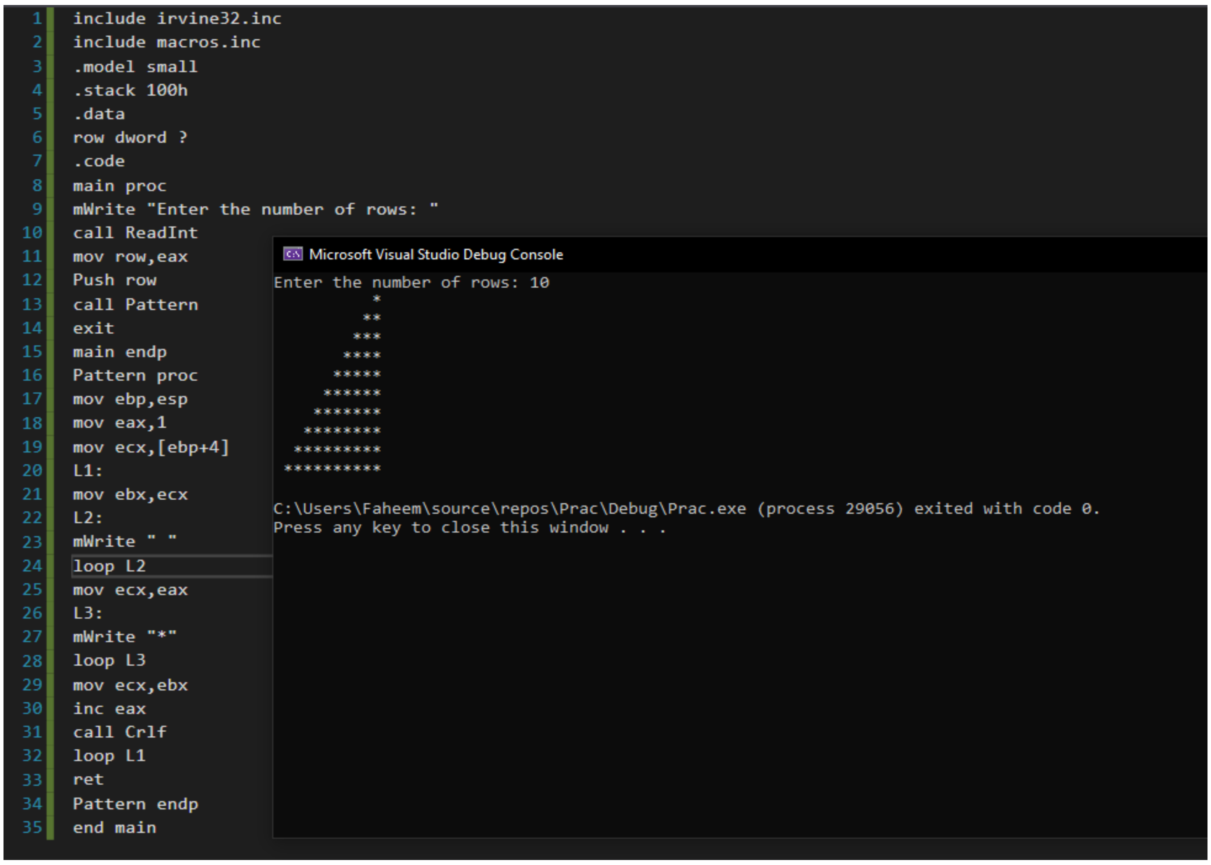


```
include irvine32.inc
include macros.inc
.model small
.stack 100h
.data
arr1 dword 10,20,30,40,50
arr2 dword 20,40,60,80,100
.code
main proc
call AddFirst
exit
main endp

AddFirst proc
mov esi,0
mov edx,0
mov ecx,5
L1:
add edx, arr1[esi]
add esi,4
loop L1
mWrite "The sum of the elements of the first array are: "
mov eax,edx
call WriteInt
call Crlf
call AddSecond
ret
AddFirst endp

AddSecond proc
mov esi,0
mov ebx,0
mov ecx,5
L2:
add ebx, arr2[esi]
add esi,4
loop L2
mWrite "The sum of the elements of the second array are: "
mov eax,ebx
call WriteInt
call Crlf
call AddBoth
ret
AddSecond endp

AddBoth proc
mWrite "The sum of the elements of both the arrays are: "
mov eax,0
add eax,ebx
add eax,edx
call WriteInt
call Crlf
ret
AddBoth endp
end main
```

Microsoft Visual Studio Debug Console

```
The sum of the elements of the first array are: +150
The sum of the elements of the second array are: +300
The sum of the elements of both the arrays are: +450

C:\Users\Faheem\source\repos\Prac\Debug\Prac.exe (process 30400) exited with code 0.
Press any key to close this window . . .
```

Task 3:

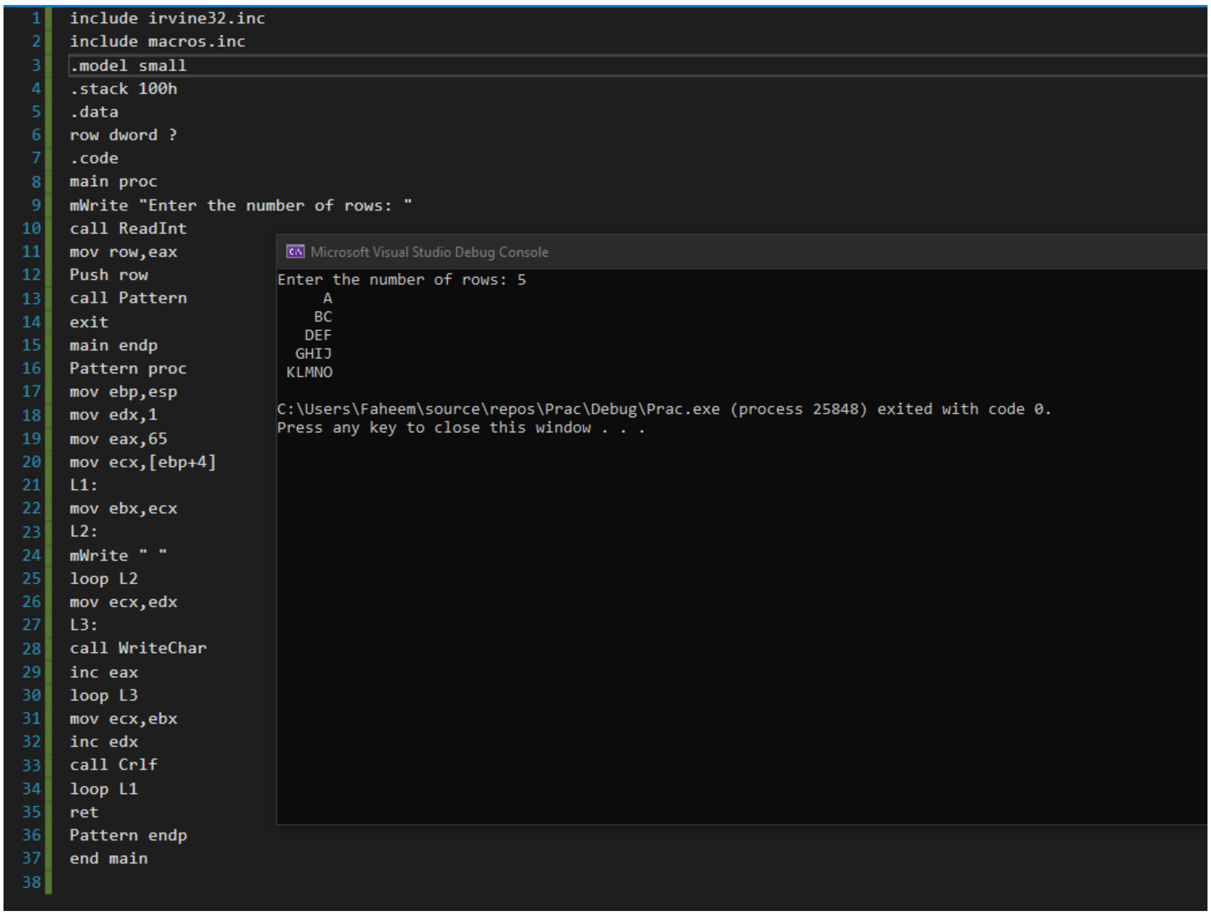


```
include irvine32.inc
include macros.inc
.model small
.stack 100h
.data
row dword ?
.code
main proc
mWrite "Enter the number of rows: "
call ReadInt
mov row,eax
Push row
call Pattern
exit
main endp
Pattern proc
mov ebp,esp
mov eax,1
mov ecx,[ebp+4]
L1:
mov ebx,ecx
L2:
mWrite " "
loop L2
mov ecx,eax
L3:
mWrite "*"
loop L3
mov ecx,ebx
inc eax
call Crlf
loop L1
ret
Pattern endp
end main
```

Microsoft Visual Studio Debug Console

```
Enter the number of rows: 10
          *
         **
        ***
       ****
      *****
     ******
    *******
   ********
  *********
 **********

C:\Users\Faheem\source\repos\Prac\Debug\Prac.exe (process 29056) exited with code 0.
Press any key to close this window . . .
```

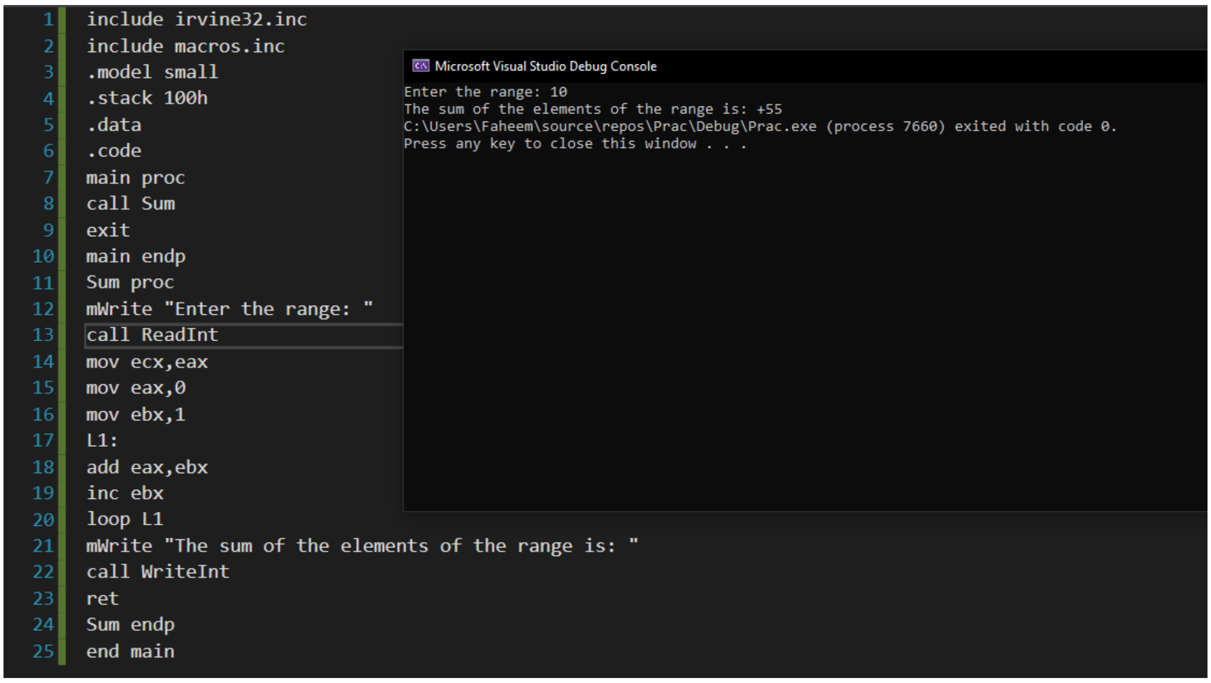
Task 4:



```
include irvine32.inc
include macros.inc
.model small
.stack 100h
.data
row dword ?
.code
main proc
mWrite "Enter the number of rows: "
call ReadInt
mov row,eax
Push row
call Pattern
exit
main endp
Pattern proc
mov ebp,esp
mov edx,1
mov eax,65
mov ecx,[ebp+4]
L1:
mov ebx,ecx
L2:
mWrite " "
loop L2
mov ecx,edx
L3:
call WriteChar
inc eax
loop L3
mov ecx,ebx
inc edx
call Crlf
loop L1
ret
Pattern endp
end main
```

```
Enter the number of rows: 5
     A
    BC
   DEF
  GHIJ
 KLMNO

C:\Users\Faheem\source\repos\Prac\Debug\Prac.exe (process 25848) exited with code 0.
Press any key to close this window . . .
```

Task 5:



```
include irvine32.inc
include macros.inc
.model small
.stack 100h
.data
.code
main proc
call Sum
exit
main endp
Sum proc
mWrite "Enter the range: "
call ReadInt
mov ecx,eax
mov eax,0
mov ebx,1
L1:
add eax,ebx
inc ebx
loop L1
mWrite "The sum of the elements of the range is: "
call WriteInt
ret
Sum endp
end main
```

```
Enter the range: 10
The sum of the elements of the range is: +55
C:\Users\Faheem\source\repos\Prac\Debug\Prac.exe (process 7660) exited with code 0.
Press any key to close this window . . .
```